NUI Galway
OÉ Gaillimh

Semester 1 Examinations 2021-22

Exam Code(s)	4BCT, 4BS, 4BMS
Exam(s)	B.Sc. (CS&IT) B.Sc. B.Sc. (Biomedical Science)
Module Code(s)	CT404, CT336
Module(s)	Graphics and Image Processing
Paper No.	
Repeat Paper	
External Examiner(s)	Dr. R. Trestian
Internal Examiner(s)	Prof. M. Madden * Dr. S. Redfern

Instructions: Answer any three questions.
All questions carry equal marks.
Each question is worth a maximum of 20 marks. The total (out of 60) will be converted to a percentage after marking.

Duration	2 hours
No. of Pages	7
Discipline(s)	Computer Science
Course Co-ordinator(s)	

Requirements:

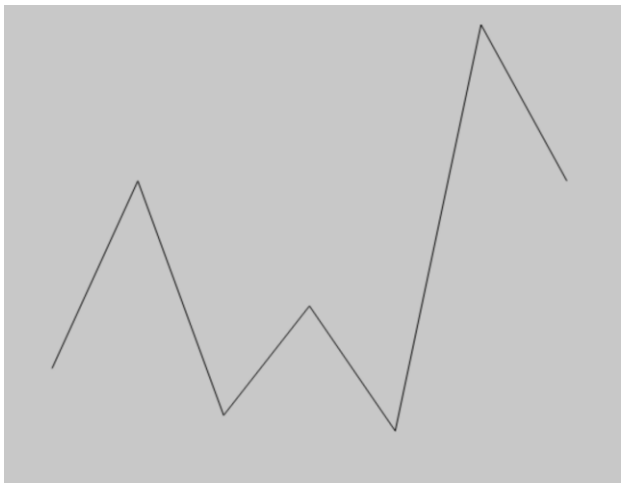
MCQ	Release to Library:	Yes	X	No
Handout				
Statistical/ Log Tables				
Cambridge Tables				
Graph Paper				
Log Graph Paper				
Other Materials				
Graphic material in colour	Yes	X	No	

PTO

Q.1. (Graphics)

(i) Provide short sections of code illustrating **translation**, **rotation**, and **scaling** in either Canvas2D or Threejs. Note that the final page of this exam paper lists some commonly used functions in Canvas2D and Threejs. [10]

(ii) Using the code below as a starting point, write Javascript/Canvas2D code for use in the **draw()** function which will draw a line graph from the numbers contained in the **data[]** array. The graph should apply appropriate scales on the *x* and *y* axes, bearing in mind that the values in **data[]** may be changed, i.e. you should not hard-code the scales. There is no requirement to label the axes. [10]



```
<html>
  <head>
    <script>
    function draw() {
        var canvas = document.getElementById("canvas");
        var ctx = canvas.getContext("2d");
        var data = [6, 18, 3, 10, 2, 28, 18];

        // to do: write code here to draw a line graph using Canvas2D

    }
    </script>
  </head>

<body onload='draw();'>
  <canvas id="canvas" width="600" height="450"></canvas>
</body>
</html>
```

Q.2. (Graphics)

(i) What are surface normals? Why are they essential to surface shading algorithms? Refer to Lambert Shading and Gourard Shading. Use diagrams to illustrate your answer. [8]

(ii) Two powerful techniques for rendering shadows in realtime 3D environments are radiosity, and ambient occlusion. Explain these techniques in simple terms, drawing attention to their suitability for pre-runtime computation. [6]

(iii) With respect to 3D graphics rendering, define the terms: specular colour, diffuse colour, ambient lighting. Illustrate each term with a diagram. [6]

Q.3. (Graphics)

(i) Many of the techniques used in real-time 3D graphics programming attempt to maximise the realism of the rendered scene while processing a minimal number of polygons. With specific reference to the so-called 'polygon budget', and using diagrams where appropriate, discuss each of the following five techniques: [15]

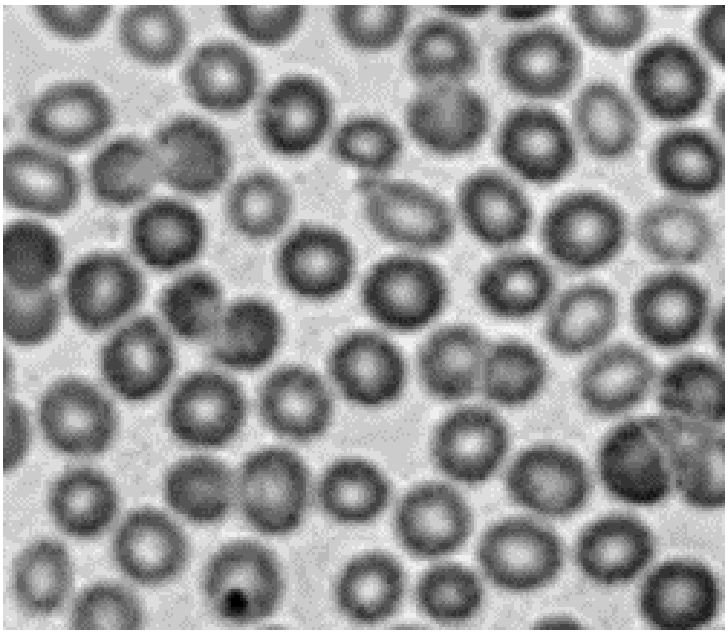
a) Frustum Culling
b) Bump Mapping
c) Back Face Culling
d) Billboards
e) Levels-of-Detail (LODs)

(ii) Discuss the Flat Shading, Gourard Shading and Phong Shading algorithms, illustrating each with a diagram. [5]

Q.4. (Image Processing)

(i) With respect to morphological image processing, outline the following operations: erosion, dilation, opening, and closing, as applied to binary images. [8]

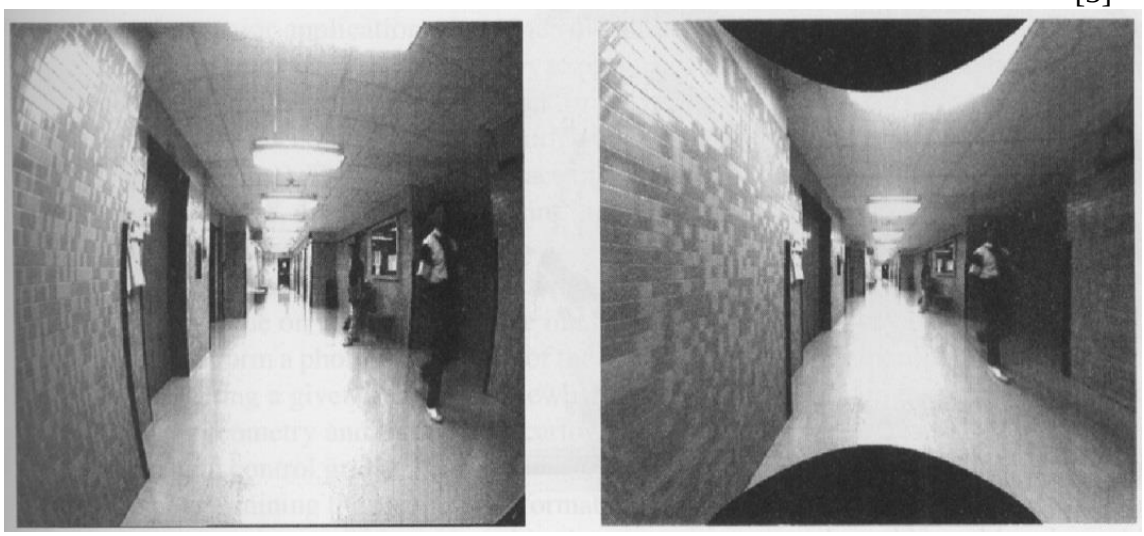
(ii) The image below is of blood cells, and it is required that a fully automated system is developed to accurately count the number of cells in images such as this. Present a suitable and robust set of image processing algorithms for this task. Explain why each step you have chosen is appropriate. [12]



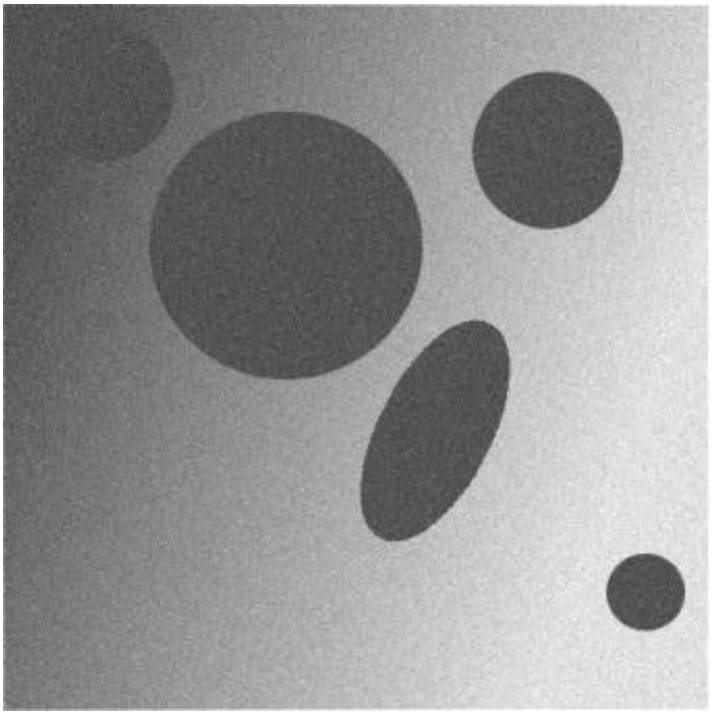
Q.5. (Image Processing)

(i) Camera decalibration is a technique for geometric correction of images which is often employed when sources of geometric error are poorly understood. With regard to camera decalibration:

- Outline the use of reference images such as grids of dots to construct and apply geometric corrections to images captured with a wide-angle lens (e.g. the image below). Use the terms 'control points', 'pixel filling', and 'bilinear interpolation' in your answer. [7]
- Explain why would you expect a reference image to be constructed with black markings on a white background (or white markings on a black background) [3]



(ii) In many cases, it may not be possible to robustly segment an image based on detected edges. One solution is to apply template matching approaches such as the Hough Transform. Explain in simple terms the operation of the Hough Transform as it applies to circles, indicating why it might be a good choice for automatic extraction of circle-like shapes such as those depicted in this noisy image. [10]



Some useful methods/properties of the Canvas 2D Context object:

Method/Property	Arguments/Values	Notes
fillRect	(Left, Top, Width, Height)	Draw a filled rectangle
beginPath	None	Start a stroked path
moveTo	(X, Y)	Move the graphics cursor
lineTo	(X, Y)	Draw a line from graphics cursor
stroke	None	End a stroked path
fillStyle	="rgb(R,G,B)"	Set fill colour
strokeStyle	="rgb(R,G,B)"	Set line colour
save	None	Save the current coordinate system
restore	None	Restore the last saved coord system
translate	(X,Y)	Translate the coordinate system
rotate	(angle)	Rotate the coordinate system clockwise, with angle in radians
scale	(X,Y)	Scale the coordinate system independently on the X and Y axes

Some useful objects/methods from the Threejs library:

Object/Method	Notes
new THREE.WebGLRenderer({canvas:c})	Constructs a renderer, attached to the Canvas object c
new THREE.PerspectiveCamera(fov,aspect,near,far)	Constructs a camera, with the specified field-of-view, aspect ratio, near clipping distance, far clipping distance
new THREE.Scene();	Constructs a scene
new THREE.PointLight(0xffffff);	Constructs a white point light
object.position.set(x,y,z)	Sets an object's x,y,z position relative to its parent
object.rotation.set(x,y,z)	Sets an object's x,y,z rotation (using Euler angles) relative to its parent
object.rotateOnAxis(new THREE.Vector3(0,1,0), 0.1)	Rotates object by 0.1 radians on the y axis
object1.add(object2)	Sets object2 as a child of object1
object.parent	Obtains a reference to the parent of object
camera.lookAt(new THREE.Vector3(0,0,0));	Turns a camera object to face the world coordinate 0,0,0
new THREE.BoxGeometry(20, 20, 20)	Constructs Box geometry, with specified width, height, depth
new THREE.MeshLambertMaterial({color: 0xfd59d7})	Constructs a Lambert (Phong) material of the specified colour
new THREE.Mesh(geometry, material)	Constructs a mesh using the specified geometry and material
renderer.render(scene, camera)	Uses a renderer to draw a scene as seen by a camera, onto the renderer's Canvas